PTO 05-0641

Japanese Patent

Document No. WO 01-33351

PROCESSOR ARCHITECTURE

[プロセッサアーキテクチャ]

Author

UNITED STATES PATENT AND TRADEMARK OFFICE

Washington, D.C. November 2004

Translated by: Schreiber Translations, Inc.

Country	:	Japan
Document No.	:	PTO 2005-0641
Document Type	:	Patent
Language	:	Japanese
Inventor	:	Toru Tsuruta, Norichika Kumamoto, Hideki Yoshizawa
Applicant	:	Fujitsu Limited
IPC	:	G06F 9/46
Application Date	:	October 29, 1999
Publication Date	:	May 10, 2001
Foreign Language Title	:	プロセッサアーキテクチャ
English Title	:	PROCESSOR ARCHITECTURE

(12) International Application Published Based on Patent Cooperation Treaty

(19) Intellectual Property Rights Agency International Bureau

(43) International Publication Date (10) International Publication No.
May 10, 2001 (10.05.2001) PCT WO 01/33351 A1

---

(51) International Patent Classification: G06F 9/46, 9/38, 15/16, 1/04

(21) International Application No.: PCT/JP99/06030

(22) International Application Date: October 29, 1999 (29.10.1999)

(25) International Application Language: Japanese

(26) International Publication Language: Japanese

(71) Applicant (All designated countries except US) : Fujitsu, Ltd. (FUJITSU LIMITED) [JP/JP];
4-1-1 Kamikodanaka Nakabara-ku Kawazaki shi Kanagawa 〒 211-8588 (JP)

(72) Inventor: [Illegible]
(75) Inventor/Applicant (Only for the US): Toru TSURUTA, Norichika KUMAMOTO, Hideki YOSHIZAWA;

Fujitsu, Ltd.
4-1-1 Kamikodanaka Nakabara-ku Kawazaki shi Kanagawa 〒 211-8588 (JP)

(74) Representative: Tadahiko ITOH;

Ebisu Garden Plaza Tower- 32nd Floor
4-20-3 Ebisu
Shibuya-ku, Tokyo 〒 150-6032

(81) Designated Country (domestic): JP; US
Attached Publication Document Type--International Study Report

Reference "Guidance Note of Code Abbreviations", for 2 character codes and other abbreviations, which is contained on the opening page of every PCT gazette which is published for the period.

---

(54) Title: PROCESSOR ARCHITECTURE

(54) Title of the Invention: Processor Architecture

[Translator's Note: Diagram on front page has notations that are already translated]

Front Page

[Translator's Note: Abstract is already translated on the Front Page and is reproduced here for the convenience of the reader]

(57) Abstract: A processor architecture comprising a program counter for executing M independent program streams on an instruction basis in timesharing manner, an N-stage pipeline shared by the program streams and operable at a frequency F, and a unit for executing the only S program streams in accordance with the operation performance required. The processor architecture is built up of M processors having an operating frequency of F/M connected in parallel, where M and N are mutually independent integers that are equal to or greater than 1, S is an integer that is equal to or greater than 0 and satisfies $S \leq M$ and N/M is the number of stages viewed from each program stream.

Abstract Page

SPECIFICATION

Processor Architecture

Technology Domain

This invention is especially concerned with the processor architecture of a multiple stage pipeline construction as it relates to processor architecture.

Background Technology

Recently, many processors have multiple stage pipeline structures with high latency which carry out commands, and assuming a one cycle throughput, a high calculation performance is realized. Moreover, when the throughput is one cycle, and because commands become equivalent that can be carried out in fractions of the operating frequency (MHz) in one second, the pipeline is partitioned and a method is used to make the delay time of part of the 1 stage smaller.

In Figures 1 and 2 is a diagram which explains the method of partitioning the processor pipeline of the processor. Within Figures 1 and 2, (a) shows the structure of the multiple-stage pipeline, and (b) shows the command latency. P1~PN and p1~pn shows the pipeline in Figures 1 and 2 (a), and A~F shows 1 program stream. In addition, in Figures 1 and 2 (b), the horizontal axis is the pipeline, and the vertical axis is the time.

Figure 1 presents the case when there are N pipeline stages, the operating frequency is 1/T, and the calculations performance is 1, and the command latency is N cycles. On the other hand, Figure 2 presents the case when the pipeline has twice as many stages as the pipeline in Figure 1 (or, the pipeline cycle is ½). For the case of Figure 2, the number of pipeline stages is 2N, the operational

frequency is 2/T, and the calculation performance is 2, and the command latency is 2N cycles.
However, when executing conditional branching commands for processors that have multiple stage pipeline construction, although several commands directly following the branching command do not branch, they certainly will execute, and the number of these commands are proportional to the number of stages of the pipeline. In this invention's specification, this reduction is called "delayed jump", and the number of commands which are executed are called the "delay number."

The reason that this delayed jump becomes disadvantageous lies with the several commands which directly follow the branch command, and even if the software developer describes with an assembler, the probability of fashioning an efficient command is low, since when having developed and using a high level language like C, and depending on a compiler, the probability of fashioning an efficient command seem smaller. Thus, in referring to an efficient command which cannot be fashioned, there results in the fashioning of a no operation (NOP: No Operation) which is an inefficient command, and as a result, there appears a cycle which cannot execute calculations, and the processor's effective performance is reduced. To rephrase, when there is an increase in the number of pipelines, the number of delays of the delayed jump increases, and because there is an increase in the cycles in which an effective command cannot be fashioned, effective command code becomes non-producible.
With an attempt at making the command code most appropriate, there is a scarcity of advantageous methods for the number of pipeline stages, but if the number of pipeline stages is increased, because it is possible to increase the operating frequency, and considering a trade-off between the former and the latter, the bulk of processors use the latter. Furthermore, because there is a limit to pipeline partitioning, the method for improving operating frequency so as to increase the performance of recent processors tends to demands of improvement in operating speed in accordance with the progress of device technology.
Thus, the number of delays of the delayed jumps becomes smaller, and continuing to strive for the most suitable of the command codes, realization of high performance processors is required. In learning from the above-mentioned problems and requirements, a high performance digital signal processor (DSP) , for example, "Pipeline

Interleaved Programmable DSP's: Architecture", IEEE Trans. Acoust., Speech, Signal Processing, Vol. 35, No. 9, Sept., 1987 was proposed by Lee et al. In this DSP architecture, concerning a DSP having a multiple stage pipeline, carrying out time-division (interleave) of the multiple program stream, in conjunction with the possibility of realizing pipeline sharing, it was reported that the effect was to reduce the number of stages of the pipeline viewed from every program stream.

Recently, there has been progress in making DSP a high performance device, and the application domain of DSP is not limited to audio processing, and is applicable even to the image processing which treats expanded information quantities. Because of this, every type of processor is required in a wide performance domain from comparatively low performance processors to processors with extremely high performance.

When it is the case of a high performance processor, naturally, the low required performance for audio processing is sufficient for execution. However, when it is the case of a high performance processor, and only the high performance portion has increased electric power consumption, and the electric power consumption, when executing the audio processing of low required performance with a high performance processor, is compared with the case of executing the same processing with a low performance processor, there is a rather significant problem.

Origin of the Invention

Thus, this invention, in executing program streams which meet the required performance, have a general goal of providing a reducible processor architecture in response to power consumption for the required performance.

The concrete goal of this invention is to provide a processor architecture with a program counter which executes using time-division in 1 command unit M independent program streams, and jointly with every program stream, and with frequency F, an operable N stage pipeline, and to provide a machine structure which executes s program streams in response to the required calculation performance, and there are integers M and N greater than or equal to 1 which are not mutually dependent, and s is an integer greater than or equal to 0 which satisfies $s \leq M$, and with the number of stages viewed from said pipeline as N/M, making a processor with an operating frequency of F/M with an M parallel structure.

In this case, for the processor architecture, for initiation of every program, it is permissible to provide beforehand a structure which dynamically executes suspension and conversion. In addition, for the above-mentioned structure, it is permissible to include a clock which is supplied to every stage of the above-mentioned pipeline, and a mask control unit which masks the assigned cycles for non-required (M-s) program streams.

Other goals of this invention are to provide a program counter which executes, using time-division for one command unit, M independent individual program streams, and together with every program stream, provide an operable N stage pipeline using F cycles, and provide a command development unit which develops one command as Q parallel commands, and execute a single individual program stream in every M cycles to respond to the required calculation performance, and provide a structure which selectively

implements Q parallel commands in the remaining (M-1) cycles, and M and N are integers greater than or equal to 1 which are not mutually independent, and Q is an integer which is greater than 1 which satisfies the relation $Q \leq M$, and the number of stages as viewed from said pipeline as seen from every program stream is considered as N/M, and with a processor architecture, having constructed M processors with a F/M operational frequency.
In this case, it is permissible to provide beforehand, as processor architecture, initiation of every program stream, and a structure which dynamically executes suspension and conversion. In addition, when assuming s to be an integer greater than or equal to 0 which satisfies the relation $s \leq M$, for the above structure, it is permissible to include a clock for every stage of the above-mentioned pipeline, and a clock control unit which masks cycles which are allocated to non-required (M-s)

program streams. Furthermore, when s is an integer greater than or equal to 0 which satisfies the relation $s \leq M$, a clock is supplied to every stage of the above-mentioned pipeline, and there is supplied, only when continuously executing, the above-mentioned Q individual parallel commands in the allocated cycles in the non-required (M-s) individual program streams, and it is permissible to have high speed execution locally as an execution structure.
With the above-mentioned processor architecture, for the above-mentioned every pipeline stage of a pipeline, it is permissible to have a structure which includes memory elements having a mode which maintains in memory input data and a mode which bypasses the input and outputs.
Furthermore, among the other goals of this invention are to have an operable N stage pipeline with operating frequency F, and when executing an individual program stream, together with inputting commands in every S cycles for the required calculation performance, and to provide a structure which masks the clock which is supplied in said pipeline in the remaining cycles for which the command does not input, and to have N and S as integers greater than or equal to 1 which are mutually independent, and having N/S as the number of stages of the viewed said pipeline from every program stream, and have a processor architecture which a processor with an operable frequency F/S.
In this case, there is included a memory element with a mode that maintains in memory the input data, and a mode, bypassing the input data, which outputs, and for the above-mentioned structure, it is permissible to mask the clock

supply to a memory element within the pipeline which can combine with the previous pipeline stage.
Moreover, for the above-mentioned processor architecture, the access latency is L cycles, and together with an operable frequency F, contains memory for which pipeline continuous access is possible, and it is permissible that memory access latency for one program stream is L/M, with L $\geq$ 1.

Furthermore, for every processor architecture mentioned above, for the above-mentioned pipeline, access latency is L cycles, and there is included memory for which pipeline continuous access is possible, and includes M independent units for every program stream, with $L \geq 1$.
Speaking of this invention, there is execution of the program stream which satisfies the required performance, and it is possible to realize a reducible processor architecture which satisfies the demands of electric power consumption.

Simple Explanation of the Drawings

Figure 1	Diagram explains former pipeline allocation method
Figure 2	Diagram explains former pipeline allocation method
Figure 3	Diagram indicates first working example of processor architecture from this invention
Figure 4	Diagram explains operational cases for all program streams
Figure 5	Diagram explains operational cases for only one program stream
Figure 6	Diagram explains the cases of M = 2 for first working example
Figure 7	Diagram explains operational conditions of program stream when M = 2
Figure 8	Diagram explains operational conditions of program stream when M = 2
Figure 9	Explains second working example of processor architecture from this invention
Figure 10	Diagram explains operating conditions for parallel commands
Figure 11	Diagram explains clock control conditions when parallel commands are operating
Figure 12	Diagram explains the third working example of the processor architecture from this invention
Figure 13	Diagram explains the fourth working example of the processor architecture from this invention
Figure 14	Diagram explains the fifth working example of the processor architecture from this invention
Figure 15	Diagram explains the sixth working example of the processor architecture from this invention
Figure 16	Diagram explains the clock operating conditions

	when program streams are operating for every S cycles
Figure 17	Diagram explains the seventh working example of the processor architecture from this invention
Figure 18	Diagram which explains when operating 2/3 pipeline stages are in bypass mode

Best Form for Implementing Invention

Below, together with diagram 3, there is an explanation of every implementation case of processor architectures which results from this invention.
In diagram 3 there is shown the first working example of processor architecture which results from this invention. The processor, which is shown in the same diagram, has program counters 11-1 ~ 11-M, selector 12, and program stream selection unit 13 and clock control unit 14.
The program stream selection unit 13 has initiators for every program stream 1 ~ M, and functions which control dynamically suspension and conversion. When initiating program streams 1~M, the program stream selection unit 13, together with supplying the program control signals in the program counters 11-1 ~ 11-M, loads the initial values which respond to this program control unit, and supplies the control signals in selector 12, and sequentially selects the program streams 1~M, supplying to the pipelines P1~PN. In addition, the program selection unit 13 executes control for the clock control unit 14. There is eliminated the clock mask which supplies the pipelines P1~PN. Moreover, M and N are arbitrary integers greater than or equal to one, and there is no dependent relationship between M and N.
When suspending pipeline streams 1~M, the pipeline stream selection unit 13 executes control for the clock control unit 14, and sets the clock mask which supplies the pipelines P1~PN. In addition, when converting the program streams 1~M, the program stream selection unit 13, together with responding to the program control signal in the program streams 11-1 ~ 11-M, loads the new values, and executes control for the clock control unit 14. The masks of the clocks that supply pipelines P1~PN are cancelled. This kind of control, according to program stream selection unit 13 performs independently for every program stream 1~M.

In this case, the number of program streams is M, and the number of pipeline stages viewed from every program stream 1~M is N/M, and the operational frequency viewed from every program stream 1~M is F/M, and the number of the processor pipeline stages is N, and the period is T, and the operational frequency of the processor is F = 1/T.

Figure 4 is a diagram which explains the operating conditions of the program stream, and shows the operational cases for all program streams 1~M. For this case, the operational period of the processor is M x T, and the command latency is M cycles. On the other hand, Figure 5 is a diagram which explains the operating conditions of the program streams, and it is a diagram which explains the cases when only one program stream is operating. Even in this case, the processor operating period is MxT, and the command latency is M cycles. Moreover, the number s of the program streams which execute in response to the required calculation performance in the processor is considered a good number, if s is an integer greater than or equal to 1 which satisfies $s \leq M$.

Thus, this invention has a multi-stage pipeline structure and implements, by time-division using single command units, the several program streams 1~M for which the program counters 11-1~11-M are independent, realizing the cooperation of the pipeline P1~PN. Because of this, together with being able to reduce to a small number the number of pipeline stages as viewed from every program stream 1~M, considering the required calculation performance, and masking the clocks of the cycles which were allocated for the operational non-required program streams, electric power savings can be realized.

When it is the case of N-stage pipelines P1~PN which can execute at the operational frequency F, and if there can be only a single program stream, according to this single program stream, the number of pipeline stages becomes N. However, in this working example, because there is time-division execution by a command unit of the M pipeline streams 1~M, as shown in Figure 4, every pipeline stream in M cycle units is executed.

From this result, every pipeline stream 1~M in M cycle units is executed, and the number of pipeline stages of every program stream 1~M can be reduced to N/M, and there can be easy realization of the most appropriate command code. Furthermore, as a parallel processing system, because it is possible to operate the operational frequency F/M processors as M units in parallel, in making a comparison when executing a unit program stream, from the multiplier effect that results from using the most appropriate command code, it is possible for there to be improvement in processor calculation performance.

In addition, when all of the calculation performance is not necessary, it is not necessary to execute all of the M program streams 1~M. Thus, in implementing only the necessary program streams in order to realize the necessary calculation performance, and in masking the clocks of the cycles which were allocated in unnecessary program streams, power savings can be realized. Thus, as shown in Figure 5, it is possible to select calculation performance and the consumption of electric power appropriate for the application.

Figure 6 is a diagram which explains the case for M = 2 in the first working example. Within the same diagram, for identical part in Figure 3 identical notation is attached, and its explanation is abbreviated. In addition, the program counters in the diagram are abbreviated.

In this case, the number of program streams is 2, the number of pipeline stages of every pipeline stream 1, 2 is N/2, and the operational frequency of every pipeline stream 1,2 is F/2, and the number of pipeline stages of the processor is N, the pipeline period is T, and the processor operational frequency is F = 1/T.

Figure 7 is a diagram which explains the operating conditions of program stream 1 when M = 2. In addition, Figure 8 explains the operating conditions of program stream 2 when M = 2. In this case, the processor's operational frequency is 2 x T, and the command latency is 2N cycles. Thus, 2 individual processors with command latency 2N cycles can operate in parallel.

Furthermore, in classifying application systems, adopting the same response as below, the most appropriate microprocessor structure can be realized.

In the first working example, when it is a case of constructing a system which necessarily requires the high calculation performance for signal processing, as shown in Figure 3, implementing every task as an independent program stream, it is possible to realize high calculation performance. In addition, because it is possible to execute independently every task, the tasks do not mutually interfere with one another and there is no reduction in executing performance.

In the second working example, when it is a case of constructing a terminal system which contains an operating system (OS), the OS is implemented in one program stream, and in the other program streams are implemented the

necessary programs, implementing multi-tasking. In addition, the cycles which were allocated to the program streams whose execution is not necessary, by masking the clocks, power savings is realized. Furthermore, as understood from Figure 5, when implementing time-division in M program streams, if the OS is implemented using 1 program stream, electric power consumption becomes approximately 1/M when all of the M program streams are executed. Furthermore, because the OS can execute freely, with the addition or elimination of tasks, even electric power consumption becomes appropriately possible to control compared to the number of tasks that are operating.

In the third working example, low calculation performance is sufficient, and when it is a case of a system which requires power savings, in the same way as with just an OS which executes using a single program stream, it is not necessary to execute all of the M program streams. Consequently, there is implementation of only a sufficient number of program streams for the purpose of satisfying the necessary calculation performance, and by masking the clocks of the cycles which were allocated to unnecessary program streams, power savings can be realized. Moreover, it is possible to select the calculation performance and electric power consumption appropriate to the application. In diagram 9 is shown the second working example of the processor architecture of this invention. For the processor, shown in an identical diagram, the program counter is 11, the command development unit 21, selector 22, the program stream selection unit 23, and the clock control unit 24. The program stream selection unit 23 has functions which control automatically the initiation, development and suspension of one program stream 1. At the time of program stream initiation, the program stream selector 23 loads the initial values corresponding to the program control signal in the program counter 11.
When developing the program stream 1, the command development unit 21 expands into Q parallel commands the 1 command of the program stream, and supplies the result to selector 22. In addition, the program stream selector 23 selects sequentially the Q parallel commands from the command development unit 21 in selector 22 and supplies to the pipeline P1~PN. It also supplies the control signal to selector 22. Furthermore, the program stream selector unit 23 executes control for the clock control unit 24, and

based on the information about the degree of command parallelism from the command development unit 21, the clock mask is set, supplying to the pipelines P1~PN.
When converting program stream 1, the program stream selector 23, together with responding to the program control signal in the program counter 11, executes control for the clock control unit 24, and eliminates the clock masks which supply the pipelines P1~PN.

Control as in program stream selection unit 23 executes for program stream 1. In this case, the number of program streams is 1, the number of pipeline stages from the point of view of program stream 1 is N/M, the operational frequency as seen from program stream 1 is F/M, and the number of processor pipeline stages is N, and the pipeline period is T, and the processor operational frequency is F = 1/T.
In this way, if this working example is considered, instead of executing using time-division the M program streams as in the first working example, there is execution of only one program stream, and during the cycles which were allocated to the remaining M-1 program streams, no commands are executed, and one command is expanded into Q parallel commands ($Q \leq M$). There is selective execution of these Q parallel commands in the remaining M-1 cycles. From this, by continuous execution of the Q cycles in the time allocated, it is possible to have locally high speed execution in a command unit.
Figure 10 is a diagram for explaining the operating conditions of parallel commands. As understood from this diagram, through execution, having embedded this command which can execute in parallel within one program stream, when the degree of parallelism is 1, operating with a processor having operational frequency of F/M, when the degree of parallelism at the command level grows, maximum M parallel execution becomes possible, and it is possible to operate the processor locally at a performance of M times. In addition, from the command development unit 21 information of the degree of command parallelism of the command unit is supplied to the clock control unit 24, and among the clocks that are supplied to the pipelines P1~ PN from the clock control unit 24, by masking the clocks of the cycles in which parallelism has been not developed, power savings is also possible. Figure 11 is a diagram

which explains the clock control conditions when command parallelism is operating.
In addition, like the third working example which is explained next, by combining the above-mentioned working examples 1 and 2, and continuing parallel execution of several program streams, it is possible to implement parallel command execution.
Figure 12 is a diagram which shows the third working example of program architecture resulting from this invention. The processor that is in the same diagram has program counters 11-1 ~ 11-M, command development unit 31, selector 32, program stream selection unit 33 and clock control unit 34. As a matter of explanation convenience, in Figure 12, when implementing parallel command execution using individual program streams, there is the assumption of executing 3 parallel commands.
The program selection unit 33 has functions which automatically control initiation, development, and suspension of the M program streams 1~M. When initiating every program stream 1~M, the program stream selection unit 33 responds to the program control signal in the program counters 11-1~11-M and loads the initial values.

When developing every program stream 1~M, the command development unit 31 expands into Q parallel commands 1 command of every program stream, and supplies these parallel commands to selector 32. In addition, the program stream selector unit 33 sequentially selects the Q parallel commands from the command development unit 31 in selector 32, and in a manner similar to supplying commands to the pipelines P1~PN, supplies the control signals to selector 32. Furthermore, the program stream selection unit 33 implements control for the clock control unit 34, and based on information about the degree of command parallelism from the command development unit 32, sets the masks of the clocks that were supplied to the pipelines P1~PN.
When converting every program stream 1~M, the program stream selection unit 33, in conjunction with loading new values in response to the program control signals in the program counters 11-1~11-M, implements control for the clock control unit 34, and based on information about the degree of parallelism from the command development unit 31, eliminates the masks of the clocks that were supplied to pipelines P1~PN.

Control in a manner according to program stream selection unit 33 is implemented for every program stream 1~M. In this case, the number of the program streams is M, the number of pipeline stages viewed from every program stream 1~M is N/M, the operational frequency as seen from every program stream 1~M is F/M, the number of processor pipelines is N, the pipeline period is T, and the operational frequency of the processor is F = 1/T.
In this way, speaking of this working example, by combining the above-mentioned working examples 1 and 2, while continuing parallel execution of the several program streams, it is possible to implement parallel command execution in every program stream. From this, for every program stream, it is possible to implement locally high speed execution in the command unit.
Figure 13 is a diagram which shows the fourth working example of the processor architecture resulting from this invention. In the same diagram, for the identical parts of Figure 3, the same notation is attached, and that explanation is abbreviated. In addition, the diagram of the program counters is abbreviated.

In this working example, as a matter of explanation convenience, M= 4, that is, assume that the number of program streams is 4. In addition, the access latency is L ($L \geq 1$), the operational frequency is F, and memory 41, which has structures for which pipeline continuous access is possible (that is, throughput is one cycle), is embedded in processor pipelines P1~PN. In addition, as a matter of explanation convenience, the number of pipeline stages of memory 41 is 4, that is, L = 4 is assumed. In this case, the number of pipeline stages of every program stream 1~4 is N/4, the operational frequency of every program stream 1~4 is F/4, the number of processor pipeline stages is N, the pipeline period is T, and the processor operational frequency is F = 1/T.
Consequently, together with being able to reduce to 1/M = ¼ the memory access latency as seen from every program stream 1~4, it is possible to share a single memory with several (M) processors.
Figure 14 is a diagram which shows the fourth working example of processor architecture resulting from this invention. Within the same diagram, identical notation is attached for the identical of Figure 3, and that explanation is abbreviated. In addition, the diagram of the program counters is abbreviated.

In this working example, as a matter of explanation convenience, M = 4, that is, one assumes that the number of program streams is 4. In addition, access latency is L cycles (L ≥ 1), the operational frequency is F/4, and one assumes that memories 43-1 ~ 43-4 and selector 44, which have structures for which pipeline continuous access is possible (that is, throughput is one cycle), are embedded in processor pipelines P1~PN. In addition, as a matter of explanation convenience, the number of the pipeline stages of every memory 43-1~43-4 is 1, that is, one assumes L = 1. In this case, the number of pipeline stages as seen from every pipeline stream 1~4 is N/4, the operational frequency as seen from every program stream 1~4 is F/4, and the number of processor pipeline stages is N, the pipeline periods are T, and the processor's operational frequency is F = 1/T.
Consequently, it is possible to reduce the memory access latency viewed from every program stream 1~4 to F/M. Because of this possibility, there can even be a reduction of the operational frequency of every memory 43-1~43-4 to 1/M = ¼, and in making a comparison with the third working example, while maintaining identical access performance, power savings can be realized.

Figure 15 is a diagram which shows the 5th working example of processor architecture resulting from this invention. Within the same diagram, identical parts relative to diagrams 3 and 9 have attached identical notation, and their explanation is abbreviated.
In this working example, the command input control unit 51 is established. This command input control unit 51, when executing one program stream, implements control by inputting the command in every S (S≥ 1) cycle. Here, S is a variable, and with register establishment, S is input into the established command input control unit 51. From this, with a match to the required performance of the program, it is possible to establish the processor's performance at 1/S. In this case, the number of the program streams is 1, the number of pipeline stages as seen from the viewpoint of the program streams is N/S, the operating frequency as seen from the viewpoint of the program streams is F/S, the number of processor pipeline stages is N, the pipeline period is T, and the processor operating frequency is F=1/T.

Figure 16 is a diagram which explains the clock control conditions when operating the program streams at every S cycle. In this case, the processor operating frequency is SxT, and the command latency is S cycles. Relative to the (S-1) cycles which do not have commands, the clock control unit is controlled by the command input control unit 51, and because it is possible to lower the operating frequency by masking the clocks which were originally required for the operation of these cycles, from Figure 16, it is understood that it is possible to realize power savings. Thus, it is possible to control electric power consumption which matches the required performance.

Figure 17 is a diagram which shows the necessary parts of the seventh working example of processor architecture resulting from this invention. Within the same diagram, parts of the diagram identical with those of Figure 15 have attached identical notation, and their explanation is abbreviated. In Figure 17, as a matter of convenience, there is shown only the structure of the stage Pi (I = 2~N-1) of the pipelines P1~PN, but other pipeline stages have similar constructions.

In Figure 17, the pipeline stage Pi has theoretical circuits 61 and 62, memory element 63, selector 64, and bypass 65. The input data from the previous pipeline stages Pi-1 passes through the theoretical circuit 61, and on the one hand, passes through the memory element 63, and on the other hand, passes through the bypass 65, supplying selector 64. The selector 64 responds to the bypass control signal, and supplies to the theoretical circuit data from the memory element 63 or the bypass 65, and the output of the theoretical circuit 62 is output to the next stage pipeline stream Pi + 1.

Thus, every pipeline stream outputs, bypassing the mode which maintains in memory the input data and bypassing the input data, giving two operating modes. The bypass mode, in order not to operate the memory element 63, masks the clocks according to the clock control unit 14 (no diagram). When executing control like inputting commands for every S cycles (S≥1), among the pipeline stages P1~PN which have N stages, the pipeline streams do not change operation even bypassing the input data, without maintaining it and outputting, that is, the pipeline streams that can combine with the previous stage's pipeline streams are dependent. In establishing the operating mode of the memory element 63

of the pipeline stream in the bypass mode according to the bypass control signal, it is possible to lower the electric power consumption for the memory maintenance operation. Thus, by using the bypass mode in one or several of the pipeline streams, it is possible to reduce the number of essential pipeline stages. Moreover, for a combination of pipeline stream it is permissible to make them continuous. Figure 18 is a diagram which explains the clock control conditions when 2/3 pipeline streams have operated in bypass mode. Thus, the identical diagram shows when there has been combination for every 3 pipeline streams. In this case, the processor operating frequency is SxT, and the command latency is S cycles. For the identical diagram, if comparing with the 6^{th} working example shown in Figure 16, there is no reduction of processor performance, and furthermore, it is understood that it is possible to lower the operating frequency.

It is possible to generate from the value of the command input cycle S and from the command input control unit 51, shown in Figure 15, the above-mentioned bypass control signal. In addition, in Figure 17, the theoretical circuits 61 and 62 are established before and after the memory element 63, but this construction can be arbitrarily changed. Furthermore, the structures of pipelines P1~PN have bypass modes and it is possible to apply them in a similar fashion as in the every above-mentioned working example.

As in the above description, referring to this invention, by executing program streams which match the required performance, it is possible to realize reducible processor architectures for electric power consumption and required performance.

From the above, this invention was explained from working examples, but this invention is not limited to the above-mentioned working examples, and it goes without saying, that various changes and improvements within this invention's field are possible.

Claims

1. A processor architecture wherein there are program counters which execute independent M program streams by time allocation in a single command unit, and wherein there are operable N stage pipelines using the operational frequency F, together with sharing using every pipeline, and wherein there is provided construction which executes only the s pipeline streams which correspond to the required performance, and wherein M and N are integers greater than or equal to 1 which are not mutually dependent, and s is an integer greater than or equal to 0 which satisfies $s \leq M$, and wherein the number of stages viewed from said pipeline which is seen from every pipeline stream is N/M, and there is constructed from M processors with operational frequencies of F/M, M parallel units.
2. A processor architecture as in claim 1 wherein there is provided additionally a construction which executes dynamically the initiation, suspension, and conversion of every program stream.
3. A processor architecture as in claim 1 or 2 wherein for the above mentioned constructions, there is included a clock control unit which masks the clocks which supply every stage of the above-mentioned pipelines and masks the cycles which were allocated in program streams of unnecessary (M-s) units.
4. A processor architecture wherein there is provided execution of a single program stream for every M cycles, corresponding to the required calculation performance, and a structure which executes selectively Q parallel commands in the remaining (M-1) cycles, and wherein there are M and N integers greater than or equal to 1 which are mutually independent and Q is an integer greater than or equal to 1 which satisfies the relation $Q \leq M$, and wherein N/M is considered as the number of stages from the point of view of said pipeline which is seen from every pipeline stream, and from M processors with operational frequency F/M, there is constructed M parallel units.
5. A processor architecture as in claim 4 wherein is provided additionally a structure which dynamically executes every pipeline stream's initiation, suspension, and conversion.
6. A processor architecture as in claim 4 or claim 5, when s is an integer greater than or equal to 0 which

satisfies the relation $s \leq M$, and wherein for the above-mentioned structure, there is included a clock control unit which masks the clocks supplying every stage of the above-mentioned pipelines, and which masks those cycles which were allocated to (M-s) program streams whose execution proved unnecessary.

7. A processor architecture as in any one claim of the claims 4~6 wherein there is continuous execution of the clocks that supply every stage of the above-mentioned pipelines, and wherein there are the above-mentioned Q parallel commands in those cycles which were allocated in the (M-s) program streams which proved unnecessary for execution, and wherein there is locally high speed execution of the commands.
8. A processor architecture in any of the claims 1~7 wherein, for every pipeline stream of the above-mentioned pipeline there is included a memory element which has a mode which maintains in memory the input data and bypassing the input data, an output mode.

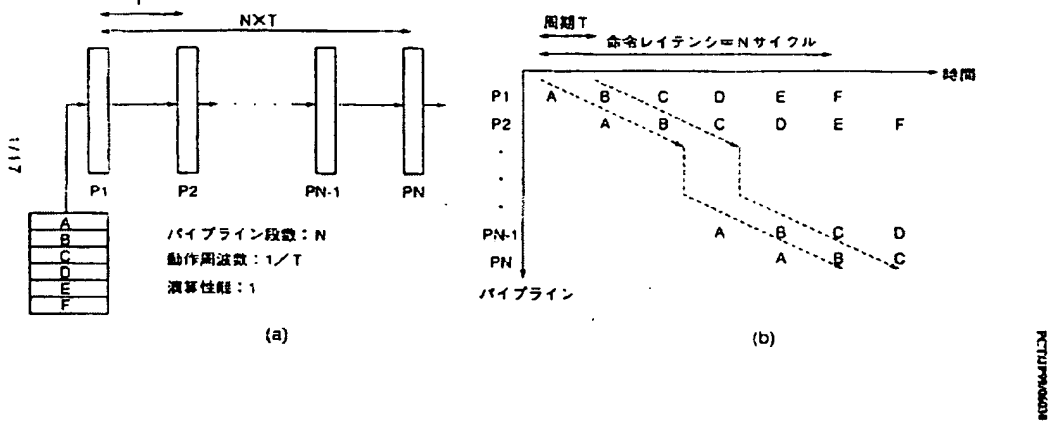
9. A processor architecture wherein there is provided an operational N-stage pipeline with operational frequency F and wherein there is provided, when there is execution of one program stream, and together with the input in every S cycle of commands which correspond to the required performance, in the remaining cycles for which commands are not inputted, a structure which masks the clocks which supply to said pipeline., and wherein there are integers N and S greater than or equal to 1 which are not mutually dependent, and wherein there is construction of processors with operational frequency F/S, with N/S as the number of stages viewed from said pipeline which in turn is seen from every program stream.
10. A processor architecture as in claim 9 wherein, for every pipeline stream of the above-mentioned pipeline, there is included a memory element which has a mode which maintains in memory the input data and which has a mode which bypasses the input mode and outputs, and for the above-mentioned structure, masks the supply of clocks to the memory element within the pipeline stage which can combine with the pipeline stream of the previous stage.
11. A processor architecture as in any one claim of the claims from 1~10 wherein for the above-mentioned pipeline, the access latency is L cycles, and together with an operational frequency F, includes a memory

with possible pipeline continuous access, and memory access latency of L/M in one of the program streams, and with $L \geq 1$.

12. A processor architecture as in any one of the claims 1~10 wherein for the above-mentioned pipeline, the access latency is L cycles, and there is included memory with possible pipeline continuous access, and M independent units corresponding to every program stream.

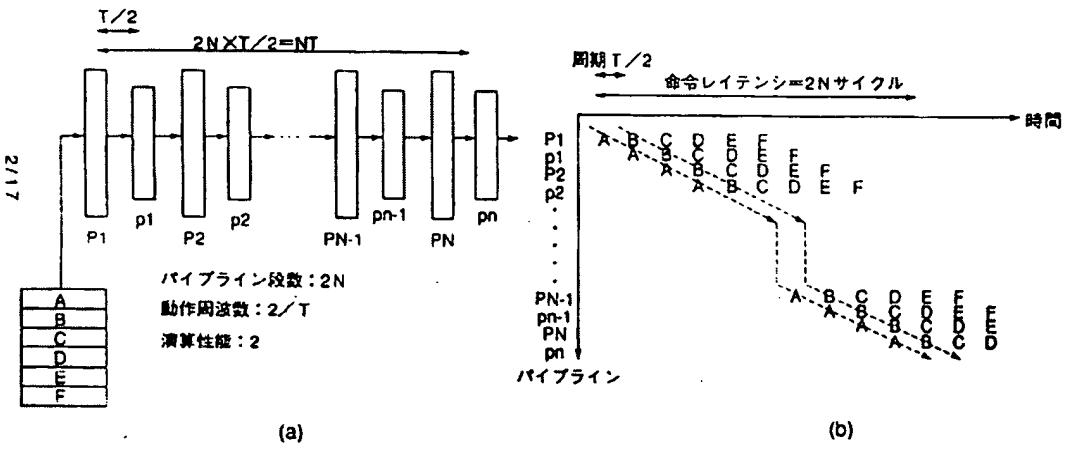
Figure Notations

Fig.1



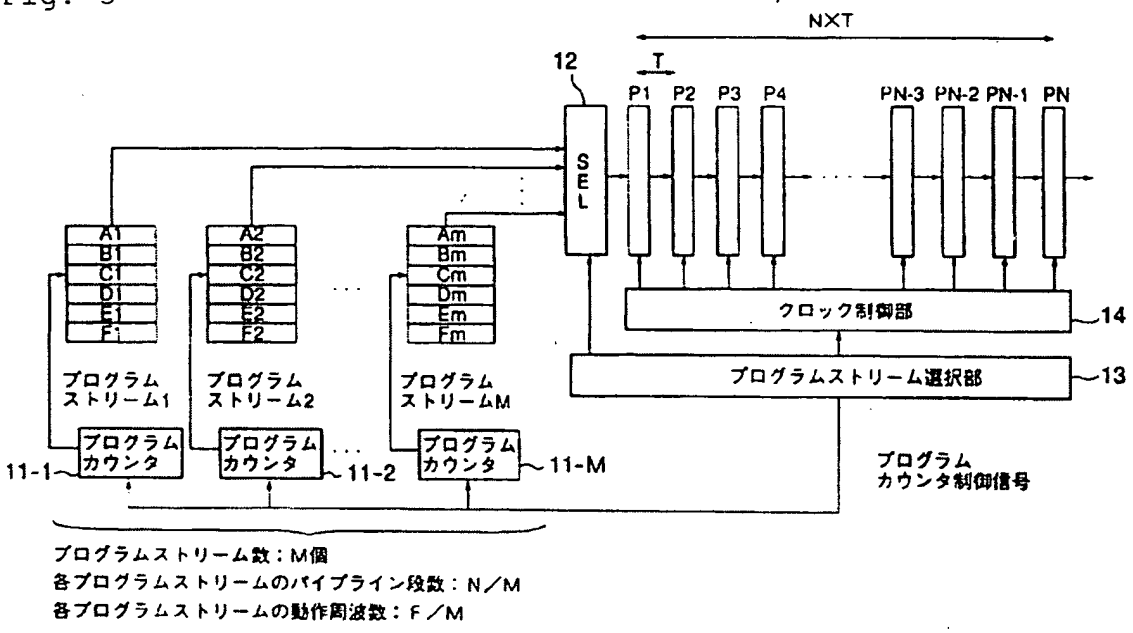
1) Number of pipeline stages: N
2) Operational frequency: 1/T
3) Calculation performance : 1
4) Period T
5) Command latency = N cycles
6) Time
7) Pipeline

Fig.2



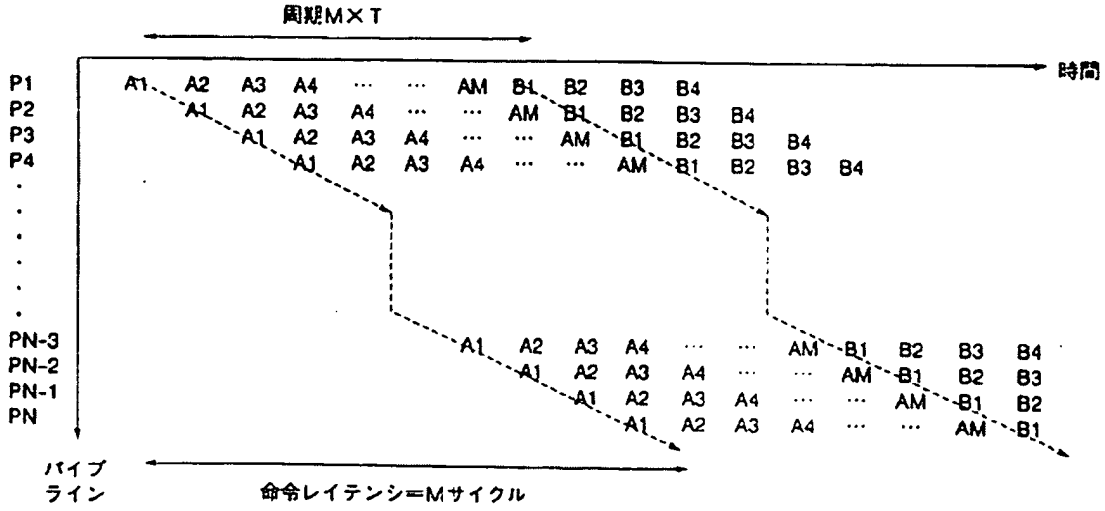
1) Number of pipeline stages: N
2) Operational frequency: 2/T
3) Calculation performance: 2
4) Period T/2
5) Command latency = 2N cycles
6) Time
7) Pipeline

Fig. 3



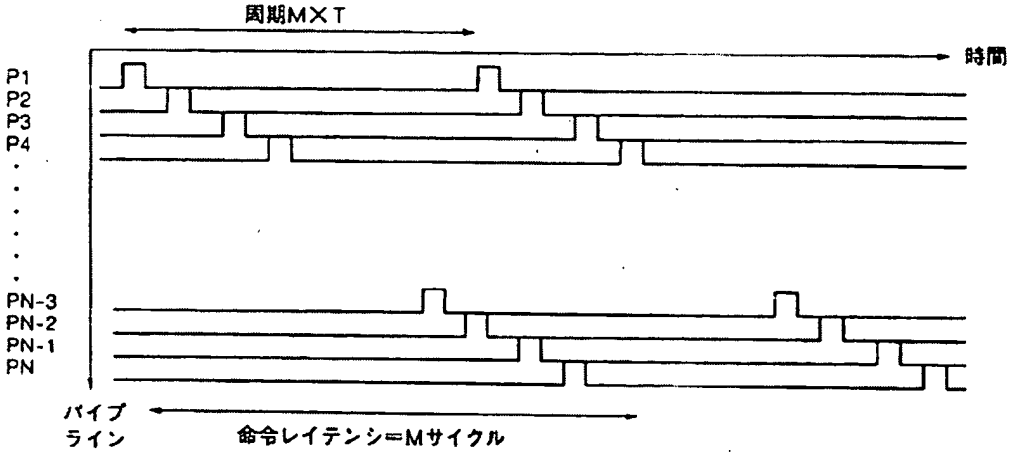
1) Number of pipeline stages of processor: N
2) Processor's operational frequency: F = 1/T
3) Program stream 1
4) Program stream 2
5) Program stream M
6) Program counter
7) Program counter
8) Program counter
9) Number of program streams: M
10) Number of pipeline stages of every program stream: N/M
11) Operational frequency of every program stream: F/M
12) Clock control unit
13) Program stream selection unit
14) Program counter control signal

Fig. 4



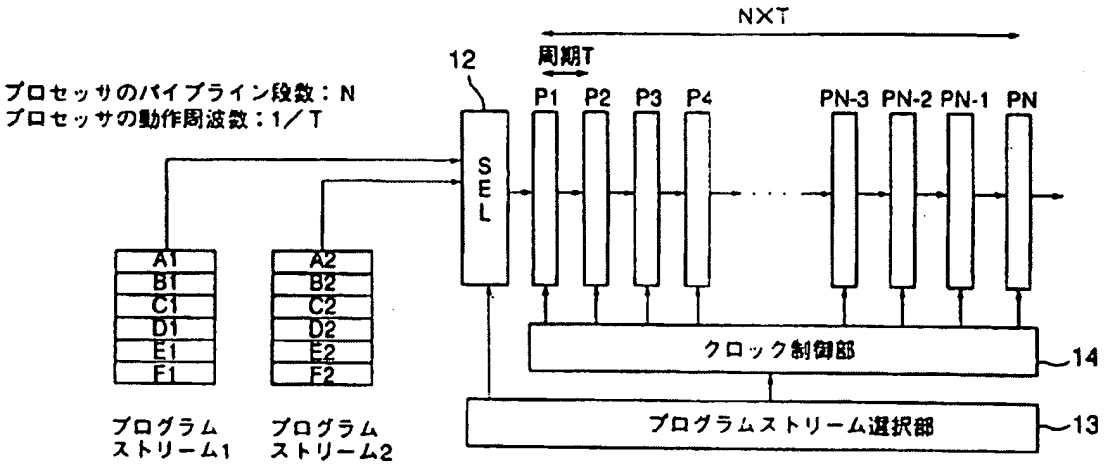
1) Period MxT
2) Time
3) Pipeline
4) Command latency = M cycles

Fig. 5



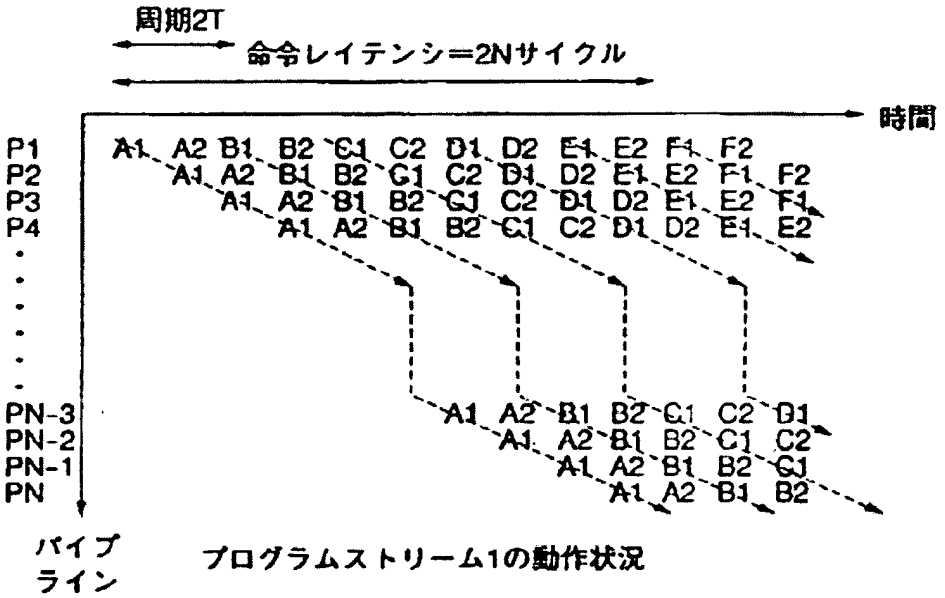
1) Period MxT
2) Time
3) Pipeline
4) Command latency = M cycles

Fig. 6



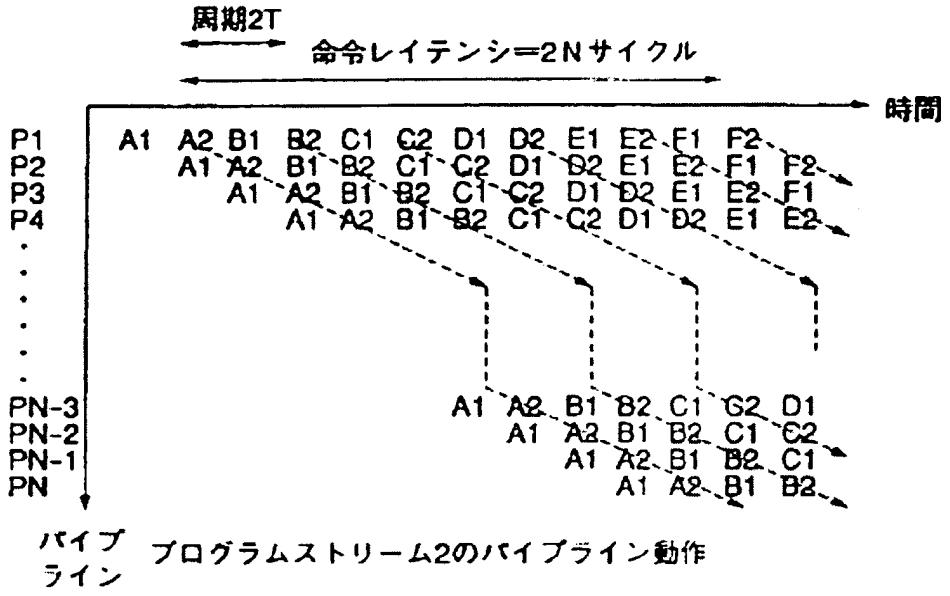
1) Number of pipeline stages of processor: N
2) Processor's operational frequent: 1/T
3) Period T
4) Program stream 1
5) Program stream 2
6) Clock control unit
7) Program stream selection unit

Fig. 7



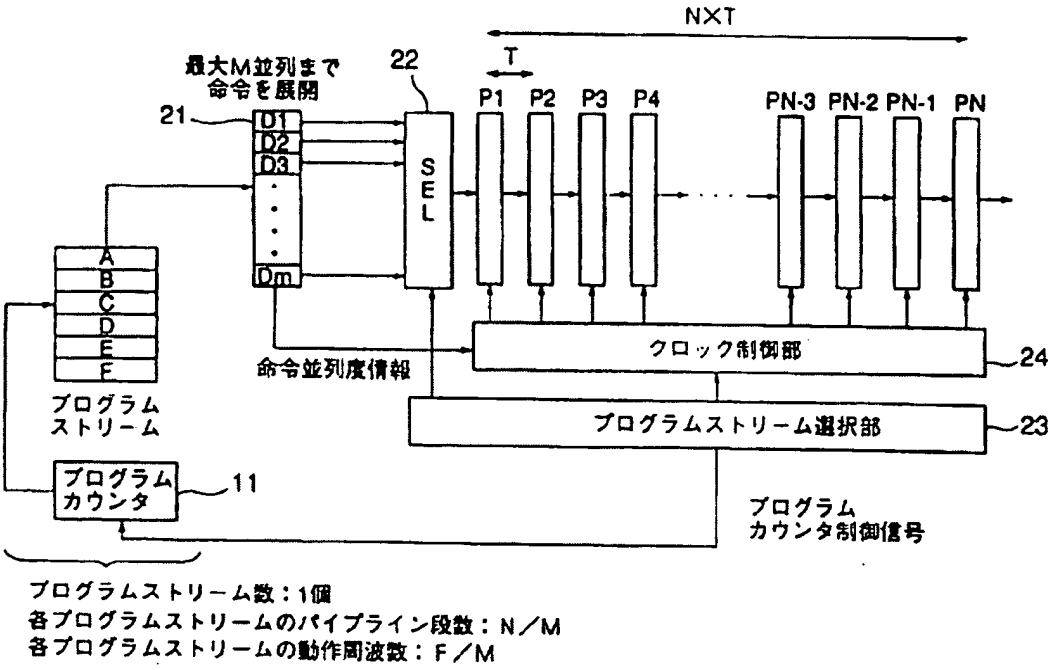
1) Period 2T
2) Command latency = 2N cycles
3) Pipeline
4) Operational conditions of program stream 1

Fig. 8



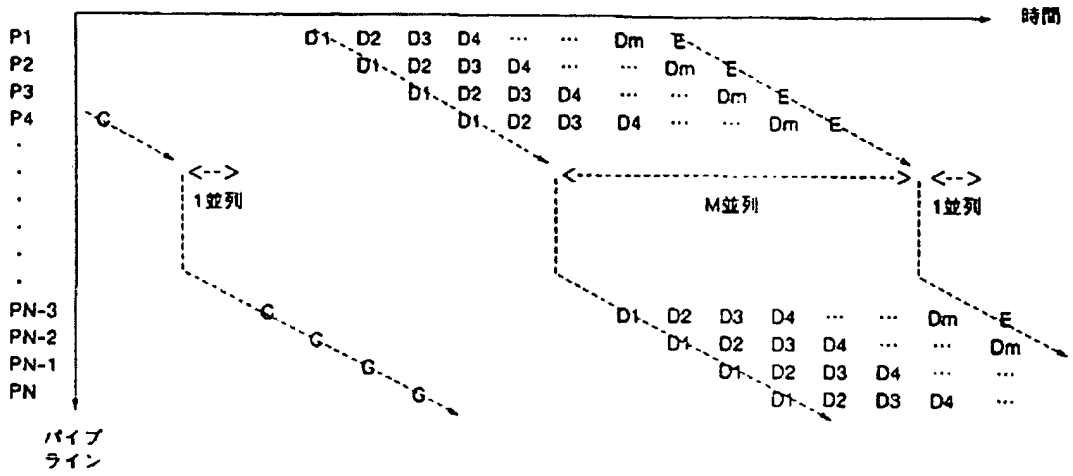
1) Period 2T
2) Command latency = 2N cycles
3) Pipeline
4) Pipeline operational conditions of program stream 2

Fig. 9



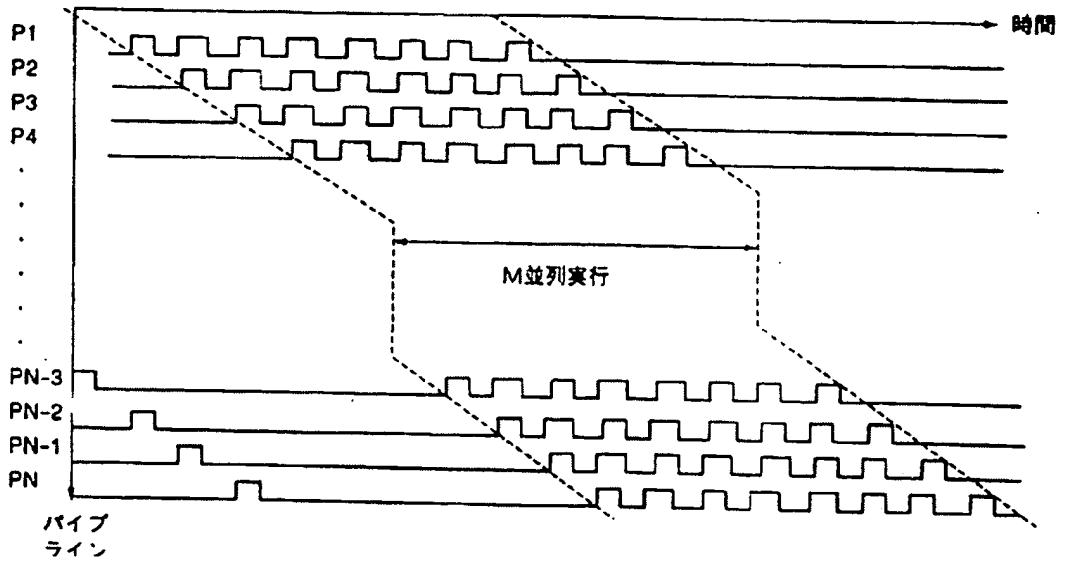
1) Number of pipeline stages of processor: N
2) Processor's operational frequency: F = 1/T
3) Development of commands up to maximum M parallel
4) Program stream
5) Information of degree of command parallelism
6) Program counter
7) Number of program streams:1
8) Number of pipeline stages of every program stream
9) Operational frequency of every program stream
10) Clock control unit
11) Program stream selection unit
12) Program counter control signal

Fig. 10



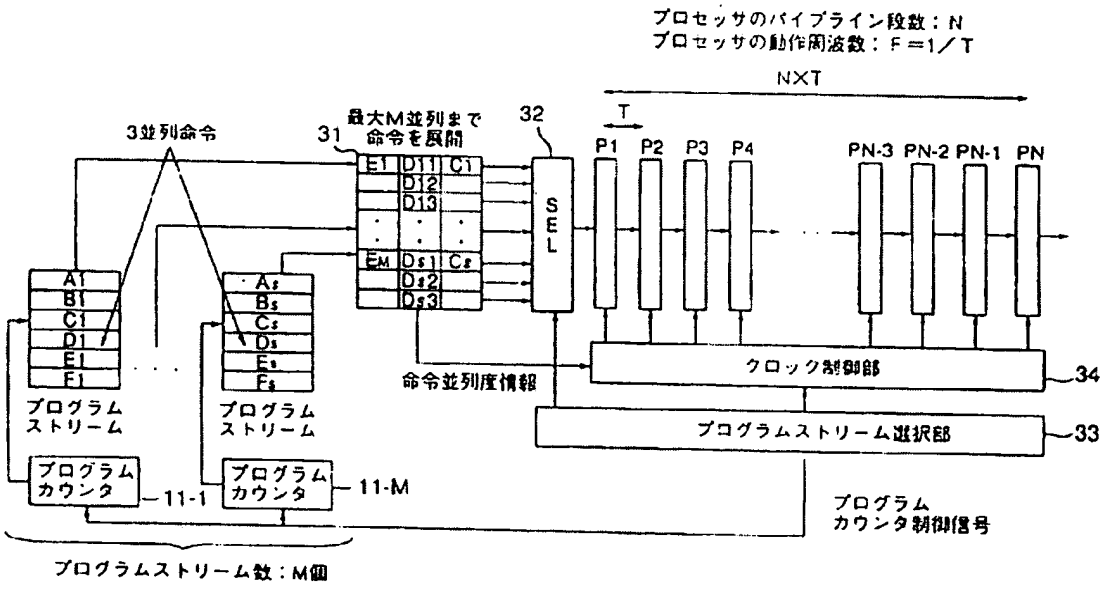
1) Time
2) 1 parallel
3) M parallel
4) 1 parallel
5) Pipeline

Fig. 11



1) Time
2) M parallel execution
3) Pipeline

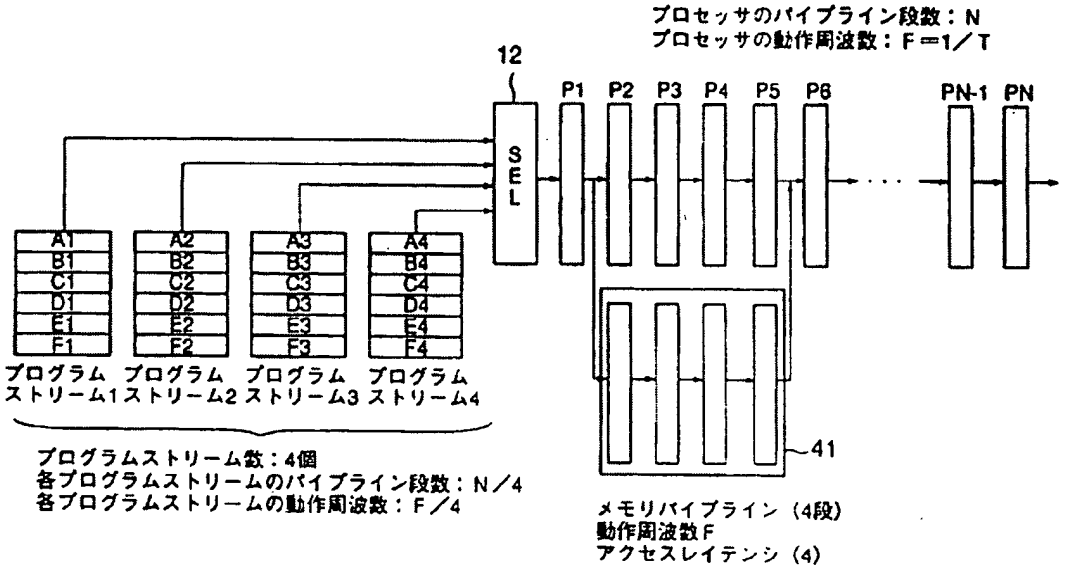
Fig. 12



1) Number of pipeline stages of processor: N
2) Processor's operational frequency: F = 1/T
3) Command development up to max M parallel
4) Program stream
5) Program stream

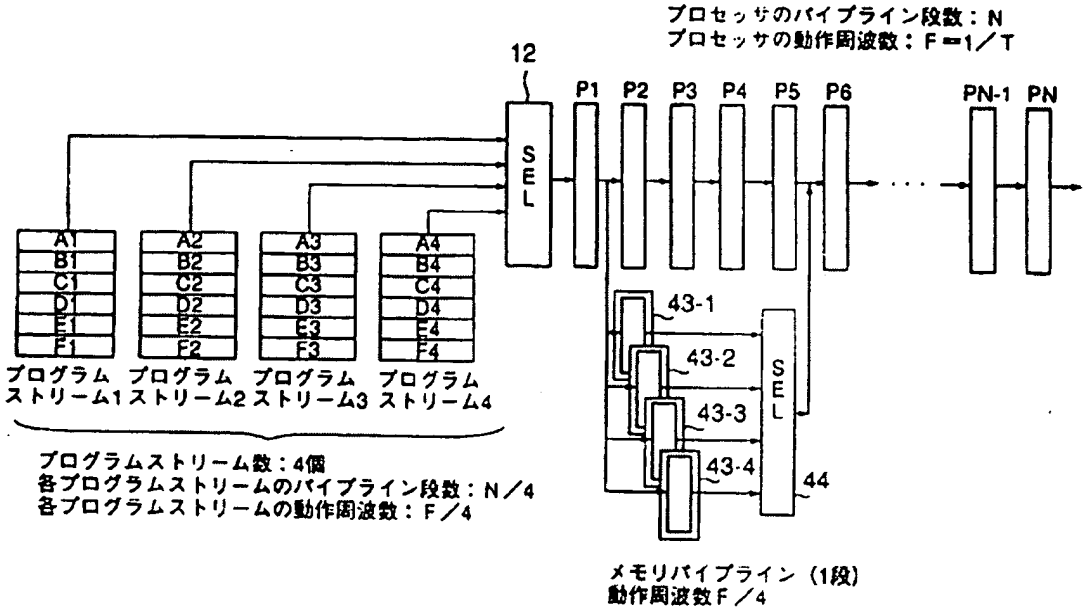
6) Program counter
7) Program counter
8) Information on degree of parallelism
9) Clock control unit
10) Program stream selection unit
11) Number of program streams: M
12) Number of pipeline stage for every program stream: N/M
13) Operational frequency for every program stream: F/M
14) Program counter control signal

Fig. 13



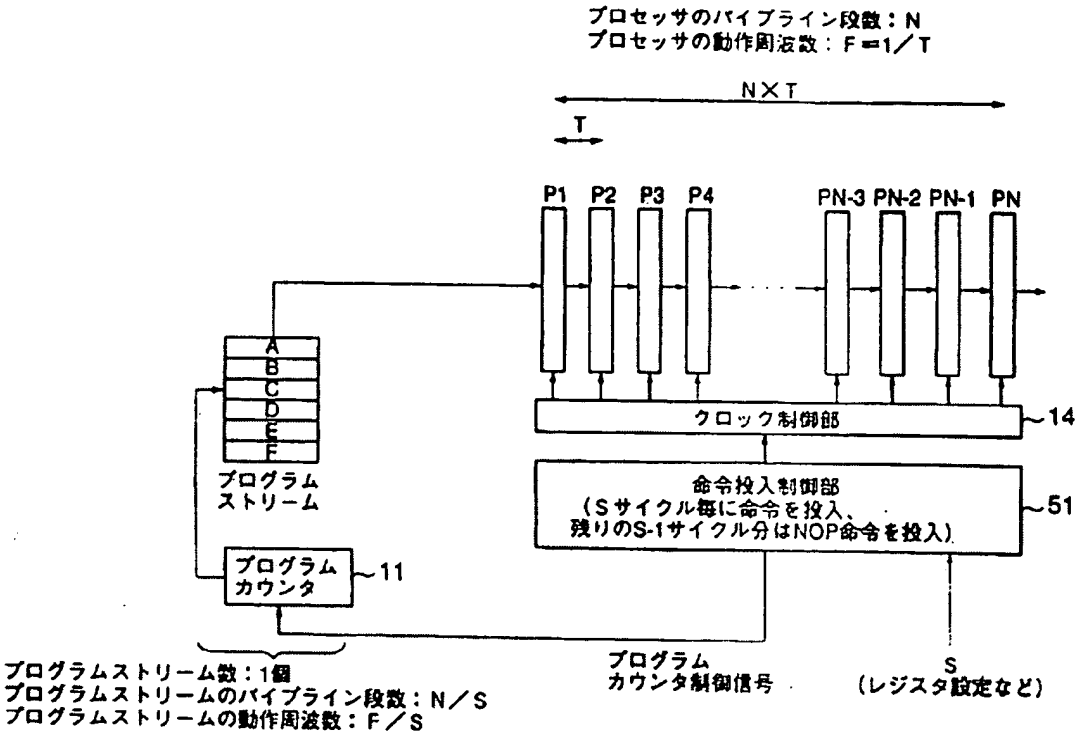
1) Number of pipeline stages for the processor: N
2) Operational frequency of processor: F = 1/T
3) Program stream 1
4) Program stream 2
5) Program stream 3
6) Program stream 4
7) Number of program streams: 4
8) Number of pipeline stages for every program stream: N/4
9) Operational frequency for every program stream: F/4
10) Memory pipeline (4 stages)
11) Operational frequency F
12) Access latency

Fig. 14



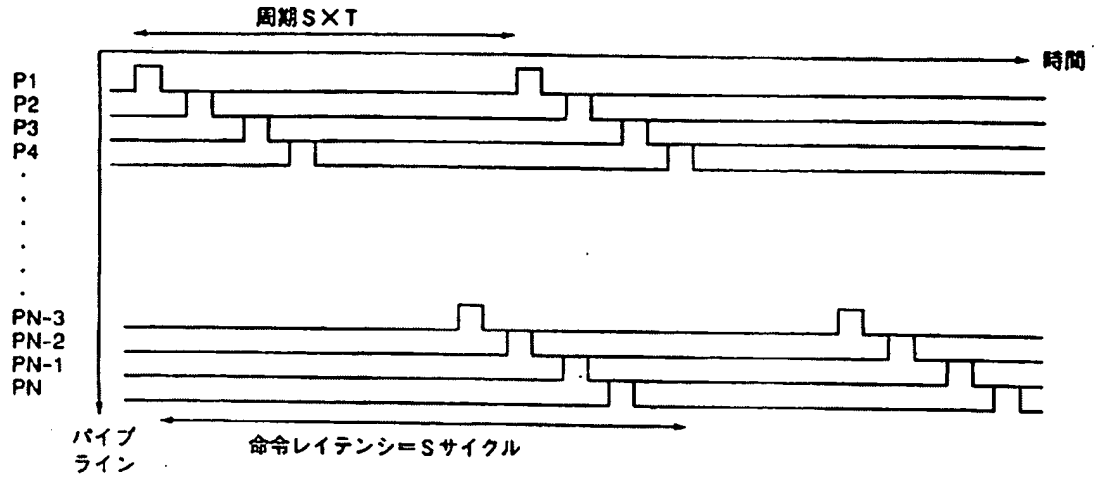
1) Number of pipeline stages of the processor: N
2) Operational frequency of the processor: F = 1/T
3) Program stream 1
4) Program stream 2
5) Program stream 3
6) Program stream 4
7) Number of program streams: 4
8) Number of pipeline stages for every program stream: N/4
9) Operational frequency for every program stream: F/4
10) Memory pipeline)1 stage)
11) Operational frequency F/4

Fig. 15



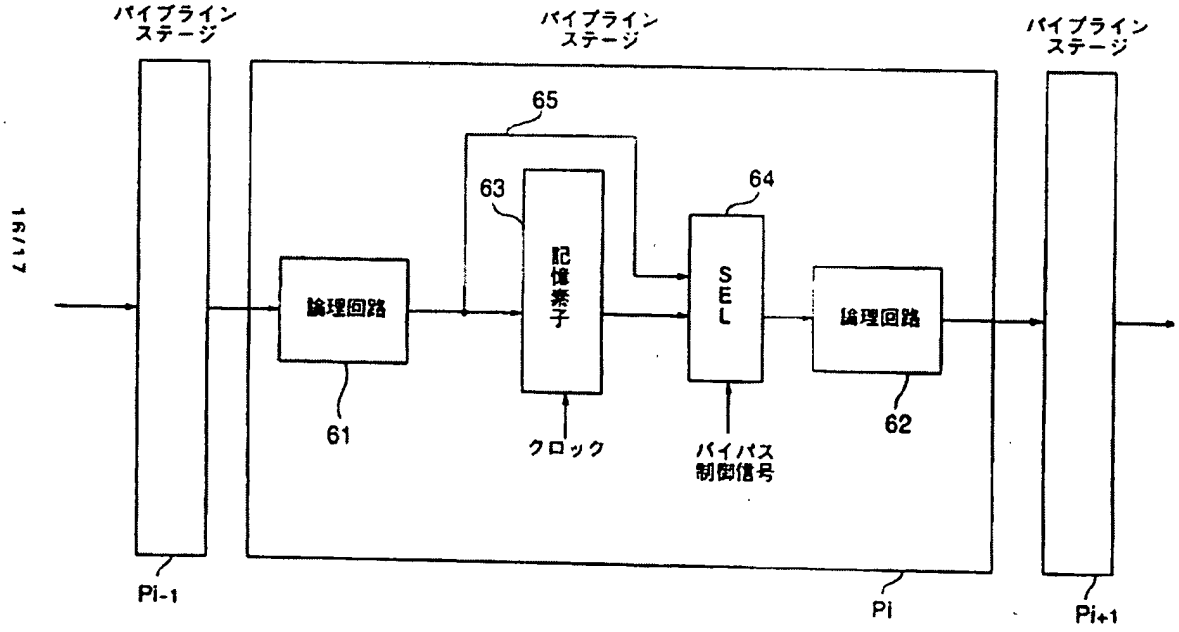
1) Number of pipeline stages of the processor: N
2) Operational frequency of the processor
3) Program stream
4) Program counter
5) Number of program streams: 1
6) Number of pipeline stages for program stream: N/S
7) Operational frequency of the program stream: F/S
8) Clock control unit
9) Command input unit (input command for every S cycles, and input NOP command for remaining S-1 cycles)
10) Program counter control signal
11) S (register establishment)

Fig. 16



1) Period SxT
2) Time
3) Pipeline
4) Command latency = S cycles

Fig. 17



1) Pipeline stage
2) Pipeline stage
3) Pipeline stage
4) Theoretical circuit

5) Memory element
6) Clock
7) Bypass control signal
8) Theoretical circuit

Figure 18